Characterization of P2P Systems

Daniel Stutzbach and Reza Rejaie

1 Introduction

Understanding existing systems and devising new P2P techniques relies on having access to representative models derived from empirical observations of existing systems. However, the large and dynamic nature of P2P systems makes capturing accurate measurements challenging. Because there is no central repository, data must be gathered from the peers who appear and depart as users start and exit the P2P application. Even a simple task such as counting the number of peers can be challenging since each peer can only report its immediate overlay neighbors.

The first half of this chapter surveys techniques for measuring attributes of P2P systems as well as characterizations derived from the application of those techniques. The second half explores two measurement techniques in detail—crawling and sampling—and demonstrates the importance of validating measurement methodology.

Systematically tackling the problem of characterizing P2P systems requires a structured organization of the different components. At the most basic level, a P2P system consists of a set of connected peers. We can view this as a graph with the peers as vertices and the connections as edges. One fundamental way to divide the problem space is into *properties of the peers* versus *properties of the way peers are connected*. Another fundamental division is examining the way the system *is* versus the way the system *evolves*. In some sense, any property may change with time and could be viewed as system evolution. We use the term "static properties" to refer to properties that can be measured at a particular moment in time and modeled with a static model (e.g., peer degree), and the term "dynamic properties" to refer to properties that are fundamentally dynamic in nature (e.g., session length). Table 1 presents

Daniel Stutzbach
Stutzbach Enterprises, Dallas, Texas. e-mail: daniel@stutzbachenterprises.com

Reza Rejaie
University of Oregon, Eugene, Oregon. e-mail: reza@cs.uoregon.edu

an overview of several interesting properties categorized by whether they are static or dynamic, and whether they are peer properties or connectivity properties.

2 Measurement Techniques

Empirical P2P studies employ one of five basic techniques, each offering a different view with certain advantages and disadvantages:

- **Passive Monitoring**: Eavesdrop on P2P sessions passing through a router.
- **Participate**: Instrument P2P software and allow it to run in its usual manner.
- **Crawl**: Walk the P2P network, capturing information from each peer.
- **Sample**: Select a subset of the peers in the network.
- **Centralize**: Rely on logs maintained by a central server.

Table 2 summarizes the peer-reviewed studies in each category and lists the particular systems they examine. Studies which intercept data have typically focused on Kazaa, which was one of the most popular peer-to-peer systems. Saroiu *et al.* [41] show that in 2002 Kazaa traffic was between one and two orders of magni-

	Peer Properties	Connectivity Properties
Static Properties	Available resources (e.g., files) Geographic location	Degree distribution Clustering coefficient Shortest path lengths Resiliency
Dynamic Properties	Session length Uptime Remaining uptime Inter-arrival interval Arrival rate	Stable core Search efficiency Search reliability

Table 1: Groups of properties

Intercept	Participate	Crawl	Sample	Centralize
[37] (B,D,G,Z)	[17] (G)	[8] (G)	[7] (N,G)	[18] (B)
[14] (Z)	[21] (G)	[3] (G)	[42] (N,G)	[38] (B)
[23] (Z)	[22] (G)	[40] (G)	[4] (O)	[15] (B)
[24] (Z)	[35] (G)	[46] (G,K,B)	[11] (D)	[51] (*)
[41] (Z,G)	[44] (G)	[48] (G)	[13] (S)	
[43] (Z,G,*)	[2] (G)		[45] (K)	
[20] (B,D,G,Z,N,*)	[10] (B)			
	[27] (Z)			
	[28] (Z)			

Table 2: File sharing measurement studies, grouped by technique. The system under study is shown in parenthesis. B=BitTorrent, D=eDonkey 2000, G=Gnutella, K=Kad, N=Napster, S=Skype, O=Overnet, Z=Kazaa, *=Miscellaneous

tude larger than Gnutella traffic. However, others studies tend to focus on Gnutella, which has several open source implementations available and open protocol specifications. Other popular file-sharing networks such eDonkey 2000, Overnet, and Kad remain largely unstudied. Each of the different measurement techniques has different strengths and weaknesses, explained in detail below.

2.1 Passive Monitoring

Monitoring peer-to-peer traffic at a gateway router provides useful information about dynamic peer properties such as the types and sizes of files being transferred. It also provides a limited amount of information about dynamic connectivity properties such as how long peers remain connected. However, passive monitoring suffers from three fundamental limitations, described below.

First, because it looks at only a cross-section of network traffic, usage patterns may not be representative of the overall user populations. For example, two of the most detailed studies of this type [14, 41] were both conducted at the University of Washington (UW). Because the University has exceptional bandwidth capacity and includes an exceptional number of young people, their measurements may capture different usage characteristics than, for example, a typical home broadband user. This limitation may be somewhat overcome by comparing studies taken from different vantage points. One study [43] overcomes the single-viewpoint limitation by capturing data at several routers within a Tier-1 ISP.

The second limitation of passive monitoring is that it only provides information about peers that are actively sending or receiving data during the measurement window. Monitoring traffic cannot reveal any information about peers which are up but idle, and it is not possible to tell with certainty when the user has opened or closed the application. These caveats aside, passive monitoring is quite useful for providing insight in file sharing usage patterns.

The third limitation is the difficulty in classifying P2P traffic. Karagiannis *et al.* [20] show that the most common method of identifying P2P traffic, by port number, is increasingly inaccurate.

The passive monitoring technique is predominantly used to study bulk data movement such as HTTP-like file transfers and streaming, where it is relatively easy to identify a flow at its beginning and count the bytes transferred.

2.2 Participate

Instrumenting open-source clients to log information on disk for later analysis facilitates the study of dynamic connectivity properties, such as the length of time connections remain open, bandwidth usage, and the frequency with which search requests are received. However, there is no guarantee that observations made at one vantage point are representative. Some studies employ multiple vantage points, but the vantage points still typically share common characteristics (e.g., exceptionally high bandwidth Internet connections) and still may not be representative.

2.3 Crawl

A crawler is a program which walks a peer-to-peer network, asking every known peer for a list of its neighbors to iteratively explore the entire graph, similar to the way a web-spider operates. Crawling is the only technique for capturing a full snapshot of the topology, needed for graph analysis and trace-driven simulation. However, accurately capturing the whole topology is tricky, particularly for large networks that have a rapidly changing population of millions of peers. All crawlers capture a distorted picture of the topology because the topology changes as the crawler runs.

2.4 Sample

Several studies gather data by sampling a set of peers in order to study static peer properties, such as link bandwidth and shared files. By sampling the set of peers at regular intervals, studies may also examine dynamic peer properties such as the session length distribution. To locate the initial set of peers, researchers have used techniques such as a partial crawl [4, 11, 13, 42], issuing search queries for common search terms [7, 42], and instrumenting a participating peer [7]. One drawback of sampling is that it is difficult to guarantee that the initial set of peers are representative. Additionally, when studying dynamic properties, sampling implicitly gathers more data from peers who are present for a larger portion of the measurement window.

2.5 Centralize

The final measurement technique is to use logs from a centralized source. Due to the decentralized nature of peer-to-peer networks, there typically is no centralized source. However, BitTorrent uses a centralized rendezvous point called a *tracker* that records peer arrivals, peer departures, and limited information about their download progress.

2.6 Summary

Measurement techniques for gathering data about the operation of peer-to-peer systems, summarized in Table 3, each have their advantages and disadvantages.

Technique	Advantages	Disadvantages
Passive monitoring	Provides information about traffic	May not be representative Omits idle peers Omits traffic on non-standard ports
Participate	Provides information about dynamic connectivity	May not be representative
Crawl	Captures the entire topology Unbiased	Doesn't scale May have significant distortion
Sample	Captures peer properties Unbiased techniques available	Haphazard sampling often unrepresentative Dynamic properties inherently biased toward long-lived peers
Centralize	Unbiased	Only available if system has a centralized component

Table 3: Summary of existing measurement techniques

3 What to Measure

The following subsections summarize other empirical studies of peer-to-peer systems, discuss their main findings, and identify important areas which remain unstudied.

3.1 Static Peer Properties

Saroiu, Gummadi, and Gribble provide an extensive and informative study, primarily of static peer properties [42]. While earlier work conceived of peers as equal participants, their landmark study demonstrates that in practice not all peers contribute equally to peer-to-peer systems. Using data collected from Gnutella and Napster in May 2001, their observations show a heavy skew in the distributions of bottleneck bandwidth, latency, availability, and the number of shared files for each host, with each of these qualities varying by many orders of magnitude.

Additionally, they found correlations between several of the properties. Bottleneck bandwidth and the number of uploads have a positive correlation, while bottleneck bandwidth and the number of downloads have a negative correlation. In other words, peers with high bandwidth tend to be uploading many files, while peers with low bandwidth have to spend more time downloading. Interestingly, no significant correlation exists between bottleneck bandwidth and the number of files stored on a peer.

In addition to the sweeping work of Saroiu *et al.* [42], several studies focus on examining the files shared by peers [2, 7, 11, 49]. A few results have consistently appeared in these studies. First, peers vary dramatically in the number of files that they share, with a relatively small percentage of peers offering the majority of available files. In addition, a large fraction of peers share no files at all (25% in [42], two-thirds in [2, 11], 11–13% in [49]). Second, the popularity of stored files in file-sharing systems is heavily skewed; a few files are enormously popular, while for most files only a few peers have a copy. Fessant *et al.* [11] found that it may be described by a Zipf distribution. However, Chu, Labonte, and Levine [7] found that the most popular files were relatively equal in popularity, although less popular files still had a Zipf-like distribution.

Studies also agree that the vast majority of files and bytes shared are in audio or video files, leading to the distribution of file sizes exhibiting a multi-modal behavior. Each studies shows that a plurality of files are audio files (48% in [11], 67% in [49], 76% in [7]). However, video files make up a disproportionately large portion of the bytes stored by peers (67% in [11], 53% in [49], 21% in [7]).

Fessant *et al.* [11] took the additional step of examining correlations in the files shared by peers. Their results show that users have noticeable interests, with 30% of files having a correlation of at least 60% with at least one other file. Of peers with at least 10 files in common, they found that 80% have at least one more file in common. Likewise, of peers with at least 50 files in common, in their data nearly 100% have at least one more file in common.

3.2 Dynamic Peer Properties

Most dynamic peer properties are tied to how long and how frequently peers are active. *Session length* is the length of time a peer is continuously connected to a given peer-to-peer network, from when it arrives until it departs. *Uptime* is the length of time a peer that is still present has been connected. *Remaining uptime* is how much longer until an active peer departs. *Lifetime* is the duration from the first time a peer connects to a peer-to-peer network—ever—to the very last time it disconnects. *Availability* is the percentage of time that a peer and its resources are connected to the peer-to-peer network within some window. *Downtime* is the duration between two successive sessions. Finally, an *inter-arrival interval* is the duration from the arrival of one peer until the arrival of the next peer. The session length, uptime, and remaining uptime are closely related, as shown in Figure 1. The popularity of file transfers is another dynamic peer property, which we examine separately.

Generally, the most important distribution for simulation and analysis is the session-length distribution, as it fully determines the uptime and remaining uptime distributions and strongly influences the availability. The median session length

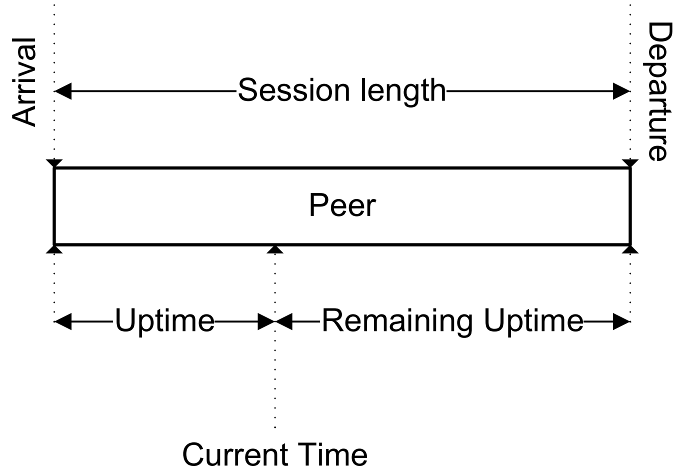


Fig. 1: Illustration of the relationship between session length, uptime, and remaining uptime

specifies how much churn a protocol must cope with, and the shape of the distribution determines whether some peers are dramatically more stable than others.

Due to its importance, several studies examine the session length distribution. Rhea, Geels, and Kubiatowicz [39] summarize these studies, as shown in Table 4 which is adapted from their paper and updated with our more recent study [46].

While the median differs dramatically, all the studies agree that the session lengths are heavily skewed: many sessions are short, while some session are very long. Several studies draw the conclusion that session lengths can be modeled with a power-law (or Pareto) distribution [5, 13, 26] or exhibit heavy-tailed behavior [13, 14, 43]. Chu, Labonte, and Levine [7] fit session lengths to a log-quadratic distribution, which can be viewed as a second-order variation of the Pareto distribution. Only one of the earlier studies [5] provide an analysis and fit to support their conclusion. However, Leonard, Rai, and Loguinov [25, pg. 8] suggest that the fit given in [5] seems implausible and point out some possible methodological errors.

Citation	Systems Observed	Session Time
[42]	Gnutella, Napster	50% $\leq$ 60 min.
[7]	Gnutella, Napster	31% $\leq$ 10 min.
[43]	Kazaa	50% $\leq$ 1 min.
[4]	Overnet	50% $\leq$ 60 min.
[14]	Kazaa	50% $\leq$ 2.4 min.
[46]	Gnutella, Kad	50% $\leq$ 15 min.–1 hr.
[46]	BitTorrent	50% $\leq$ 2 min.–30 min.

Table 4: Observed session lengths in various peer-to-peer file sharing systems. Adapted from [39].

In our more recent study [46], we examine several common methodological problems that introduce bias into studies of peer churn and examine data from Gnutella, Kad, and BitTorrent. The session lengths we observed were heavily skewed, but did not agree with a power-law or Pareto distribution. However, they could be described with a Weibull distribution.

In BitTorrent, the availability of high-quality tracker logs facilitates the study of peer dynamics. Prior studies of BitTorrent [18, 38] show that session lengths are heavily skewed. However, they do not attempt to create a model based on the data. A surprising discovery shown by Izal *et al.* [18] is that many peers (81% in their trace) depart before downloading the entire file, while peers who do complete the download linger for more than six hours on average.

While most studies of peer dynamics focus on session length, Bhagwan, Savage, and Voelker [4] provide a study of peer availability in Overnet during January 2003. However, they find that the distribution of availability varies dramatically with the size of the measurement window, due to the significant fraction of hosts who appear briefly and only once.

3.2.1 Files Transfers

Another class of dynamic peer properties is related to the files that peers are actively transferring, which in some sense is the derivative of the files being stored on each peer (discussed above under Static Peer Properties). The properties of files being transferred are most often studied using passive monitoring at gateway routers. Two of the most detailed studies of files being transfered [14, 41] were both conducted at the University of Washington (UW). The first study, [41], focuses on comparing HTTP requests with P2P requests, demonstrating that P2P uses more than twice as much bandwidth as the web on their network. Although they found a smaller number of hosts are involved in the P2P traffic, each object is orders of magnitude larger. Furthermore, they show that a majority of the P2P traffic came from a few large video files. Their second study, [14], more closely examines the popularity and properties of different P2P objects. The popularity of different objects did not match a Zipf distribution, in contrast to the Zipf distribution of popularity observed for Web objects. The authors suggest this may be due to the fact that in P2P systems, users typically download an object at most once, while Web users may return to a website many times. Instead, they found that unpopular objects appear Zipf-like, while popular objects are relatively equal in popularity, matching the results for stored files seen in [7].

Leibowitz *et al.* [23, 24] provide measurements from an Israeli Internet Service Provider (ISP), and compare their findings with the UW studies. Interestingly, they find that while the UW is an overall provider of P2P content, the ISP they study is an overall consumer of P2P content. Their studies give particular focus to the idea of caching P2P content. In [23], they implement a transparent 300 GB cache yielding a 67% bandwidth savings.

3.3 Static Connectivity Properties

In 2000, a company called "Clip2" developed a Gnutella crawler and published their results on the web. Although not validated by peer-review, their analysis and topology captures have been widely used in simulation studies of improvements for Gnutella-like networks [1, 19, 29–31]. In [8], Clip2 presents analysis of snapshots they captured between June and August of 2000. Using their crawler which took around an hour to survey the entire topology, they gathered snapshots containing between 1,000 and 8,000 peers.

Their work suggests that the Gnutella network has a power-law degree distribution, based on plotting the degree distribution of their snapshots on a log-log scale and demonstrating a linear fit. Adamic *et al.* [1] repeat this analysis on similar data provided by Clip2. However, neither study considers alternative models of the degree distribution. Several later studies [9,12,19,34,50] rely on the power-law model, simulating Gnutella using random power-law topologies. Lv *et al.* [33] show that power-law networks exhibit poorer performance than other types of random graphs.

Ripeanu, Foster, and Iamnitchi [40] implemented a crawler and use it to examine properties of the Gnutella overlay topology. Their crawler uses a client-server architecture running on roughly 50 computers to crawl a 30,000 node network in a few hours. Their crawls were conducted in November 2000 through May 2001. The size of the network grew from 2,063 to 48,195 peers over that time. They performed all-pairs shortest-path computations and plotted the distribution of path lengths. 95% of shortest-paths are 7 hops or less, with most shortest-paths being 4 or 5 hops long. They repeat the analysis of [8] and [1] by plotting the degree distribution in log-log scale. In their November snapshot, the degree distribution appears linear on the log-log plot, suggesting a power-law distribution. Their March 2001 snapshot is different. Low-degree nodes are approximately equally common, though among high-degree nodes the distribution still appears linear on the log-log plot.

Given that their crawls take a few hours, and peer uptimes may be just a few minutes [7, 14, 43], it is very possible that these topologies are highly inaccurate, leading to a drastically distorted picture of the network. In [48], we created a new crawler, Cruiser, which can crawl the Gnutella network in around 4 minutes. Later in this chapter, we provide an overview of the design of cruisers and some of the techniques we used to validate the accuracy of its snapshots. Our measurements of Gnutella were not consistent with a power-law distribution; in fact, they showed that virtually all peers had a degree under 35.

3.4 Dynamic Connectivity Properties

In [48], we explore how heavily skewed session lengths influence the topological structure. We found that long-lived peers gradually find one another and form a stable "core" for the peer-to-peer network. By remaining in the system for a long time, these peers have more opportunity to find one another. Once found, these connections remain until one of the peers leave.

When a user starts their P2P application, the application must discover other peers to form connections with. This initial discovery process is called *bootstrapping*. Karbhari *et al.* [21] provide a comparative study of the bootstrapping mechanisms of several Gnutella implementations.

Sripanidkulchai presented one of the first studies of search terms in a P2P network [44], demonstrating that queries follow a Zipf distribution, except for the most popular queries which are of roughly equal popularity (similar to the distributions of files stored and file transfers). The fact that popular queries are much more common than unpopular queries suggests caching query results may be beneficial [35, 44].

Klemm *et al.* [22] provide a comprehensive analysis of queries, breaking down the number of queries observed by time of day and geographical region. It includes distributions for the number of sessions that generate queries, the time until the first queries, the query inter-arrival time, and the length of the session. In short, it provides a framework for generating a synthetic query workload as seen from a single peer.

3.5 Summary

Peer-to-peer systems have been a popular topic for empirical studies. Existing studies cover properties of stored files, file transfers, and search terms in great detail. Additionally, Saroiu, Gummadi, and Gribble [42] provide a comprehensive study of static peer properties. However, these measurement studies have been rather ad-hoc, gathering data in the most convenient manner without critically examining their methodology for measurement bias. While an ad-hoc approach is often suitable for first-order approximations (e.g., "file popularity is heavily skewed"), it is generally not appropriate for making precise conclusions (e.g., "session-lengths are power-law"). One of the largest gaps in the existing work is the development and validation of high-fidelity measurement tools for peer-to-peer networks. The remainder of this chapter describes two tools for gathering highly accurate measurements.

4 Cruiser: a fast P2P crawler

The global state of a peer-to-peer system is distributed among all the peers. Exploring the graph and capturing the state of each peer captures the global state. A *crawler* is a tool that captures global state in this way. Since the system changes as the crawler explores, the picture is *distorted*, much like a photograph capturing rapid motion. Therefore, crawl speed is important. The faster the crawler runs, the less distortion. Crawlers have most often been used for capturing snapshots of the overlay topology as a graph, needed for studying many static connectivity properties. Many studies [3, 8, 28, 40] take 30 to 120 minutes to crawl the network and capture a snapshot of the graph. However, many peers are not even present for that long [14, 39, 43]. This suggests existing crawlers are much too slow and may be capturing very distorted snapshots. The accuracy of these snapshots most likely has not been previously addressed because it is challenging to measure the distortion without a perfect reference snapshot for comparison.

This section documents a fast crawler, called *Cruiser*, that can capture complete topologies in just a few minutes, introduces techniques for assessing the accuracy of snapshots, and shows that Cruiser may be used to capture accurate snapshots. Additional details may be found in [48]. We focus on capturing snapshots of the Gnutella topology, as Gnutella is one of the largest P2P systems and has been the target of most prior P2P crawlers, allowing us to make meaningful comparisons. Although we focus on Gnutella, Cruiser uses a plug-in architecture, allowing it to crawl other P2P systems with the addition of an appropriate plug-in.

In order to crawl quickly, the design of Cruiser must overcome several challenges:

- It must make heavy use of parallelism to contact many peers simultaneously. Managing so many connections in parallel can lead to CPU bottlenecks requiring a distributed architecture.
- If the load is too great, Cruiser may lose data.[1] Therefore, Cruiser must carefully control the load by appropriately limiting the number of connections. Since each connection uses a variable amount of resources, this limit must be dynamic.
- Cruiser cannot afford to wait the minutes for a TCP connection attempt to time-out. Instead, the proper trade-off between timing out too quickly (which increases distortion by losing data) and timing out too slowly (which increases distortion by making the crawl slower) must be found empirically.

4.1 The Design of Cruiser

Our primary goal in the design of Cruiser is to minimize distortion in captured snapshots by maximizing the speed at which Cruiser explores the overlay topology. We employ several techniques and features to achieve this design goal, as described below.

4.1.1 Two-Tier Networks

Cruiser leverages the two-tier structure of modern P2P networks by only crawling ultrapeers. Since each leaf must be connected to an ultrapeer, this approach enables us to capture all the nodes and links of the overlay by contacting a relatively small

[1] For example, connections may appear to time out if the CPU load is so great that received packets spend too long in a queue before being processed.

fraction of all peers. This strategy leads to a major reduction (around 85%) in the duration of a crawl without any loss of information.

4.1.2 Distributed Architecture

Cruiser employs a master-slave architecture in order to achieve a high degree of concurrency and to effectively utilize available resources on multiple computers. A master process coordinates multiple slave processes that act as independent crawlers and crawl disjoint portions of the network in parallel. The slaves communicate with the master using loose synchronization as follows. Each slave has an independent queue of addresses to contact, which the master fills. Each slave drains its queue by querying peers for their neighbors and reporting back with the data they've gathered. The master extracts new addresses from the data and uses this to fill the queues. The crawl terminates when all the queues are empty.

4.1.3 Asynchronous Communications

Each slave process crawls hundreds of peers in parallel using asynchronous communications. Cruiser implements an adaptive load management mechanism to ensure that slave processes remain busy but do not become overwhelmed. This is important for the steady progress of the crawl especially when different slave nodes have heterogeneous processing capabilities. Toward this end, each slave monitors its CPU load and adjusts its maximum number of parallels connections using an additive-increase multiplicative-decrease (AIMD) algorithm similar to TCP's congestion control mechanism. In practice, each PC typically runs with close to 1,000 parallel connections, contributing an additional speed-up of nearly three orders of magnitude, compared to using synchronous communications (as in [40]).

4.1.4 Appropriate Timeouts

When peers are unresponsive, waiting for TCP to timeout and give up attempting to connect takes a long time. On our systems, a full TCP timeout to an unresponsive address takes more than 3 minutes. While this is suitable for many interactive and automated applications, one of our primary design goals it to make crawls as quick as possible. We conducted an evaluation of the cost-versus-benefit tradeoff of different timeout values for crawling. As a function of the timeout length, Figure 2 shows the duration of the crawl and the percentage of peers that were unreachable. We see that while very low timeouts (less than 10 seconds) result in a dramatic increase in the number of timeouts, there are diminishing returns for using longer timeout values, while the crawl length (and thus distortion) continues to increase. In other words, if a peer has not responded after 10 seconds, it is unlikely to ever respond.

Therefore, we use a timeout of 10 seconds, providing an additional speedup of more than a factor of two, compared to using full TCP timeouts.

4.2 Quantifying Snapshot Accuracy

One obvious metric to evaluate the performance of Cruiser is the time it takes to perform a crawl. However, the crawl duration doesn't reveal how accurate the crawl is; it only informs us if the crawl is more accurate than another crawl performed under similar conditions. Snapshot accuracy can not be directly measured since there is no reference snapshot for comparison. Therefore, we must indirectly quantify the effect of crawling speed on snapshot accuracy.

To examine the impact of crawling speed on the accuracy of captured snapshots, we adjust the crawling speed (and thus the crawl duration) of Cruiser by changing the number of parallel connections that each slave process can open. Using this technique, Cruiser can effectively emulate the behavior of previously reported crawlers which have a lower degree of concurrency.

We introduce the following two metrics for evaluating a crawler. The first metric, *edge distortion*, examines the edges in the captured snapshot. For each contacted peer A, with neighbors N_A, we examine each of its neighbors $B \in N_A$ to see if they likewise reported A as their neighbor. If not, we have an inconsistency in the graph caused by the fact that the edge changed sometime between crawling node A and

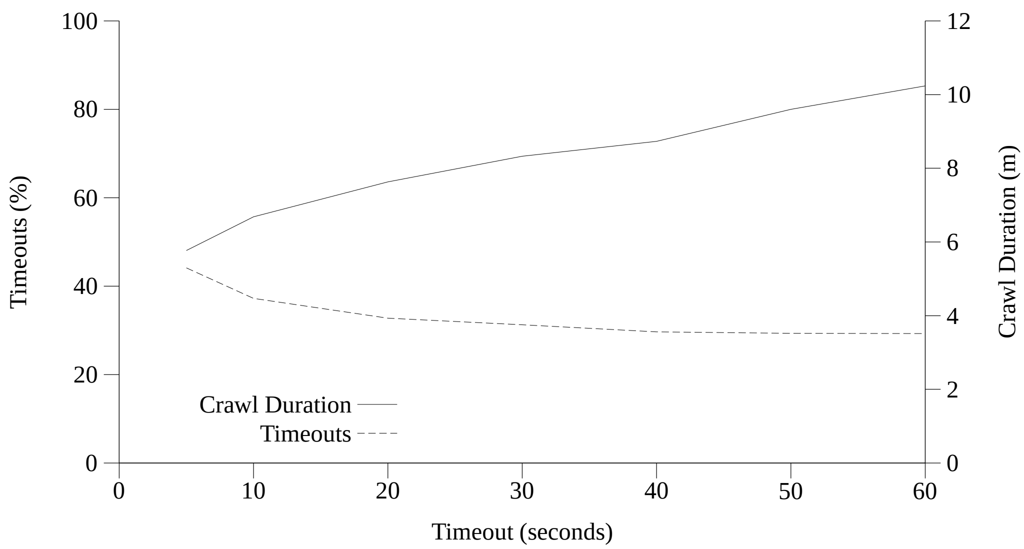


Fig. 2: Effects of the timeout length on crawl duration and snapshot completeness

crawling node B. The edge distortion, then, is the fraction of edges that are inconsistent.

The second metric, *node distortion*, examines the peers present in two consecutive snapshots captured back-to-back. We denote the peers as the sets V_1 and V_2. Comparing these two back-to-back snapshots provides insight into how distorted our picture of the network is. If Cruiser were instantly fast and captured perfect snapshots, V_1 and V_2 would be identical. The greater the change that occurs while Cruiser runs, the greater the difference between V_1 and V_2. We define the node distortion as $\frac{|V_1 \Delta V_2|}{|V_1|+|V_2|}$, where $V_1 \Delta V_2$ is the symmetric difference of V_1 and V_2 (i.e., peers in one set or the other, but not both). Note that when $V_1 = V_2$, the node distortion is 0%, and when V_1 and V_2 are completely disjoint the node distortion is 100%.

Figure 3 depicts peer and edge distortion as a function of crawl duration. This figure demonstrates that the accuracy of snapshots decreases with the duration of the crawl, because the increased distortion reflects changes in the topology that occur *while the crawler is running*. Crawlers that take 1–2 hours (comparable to those in earlier works) have a node distortion of 9%–15% and an edge distortion of 31%–48%, while at full speed Cruiser exhibits a node distortion of only 4% and an edge distortion of only 13%.

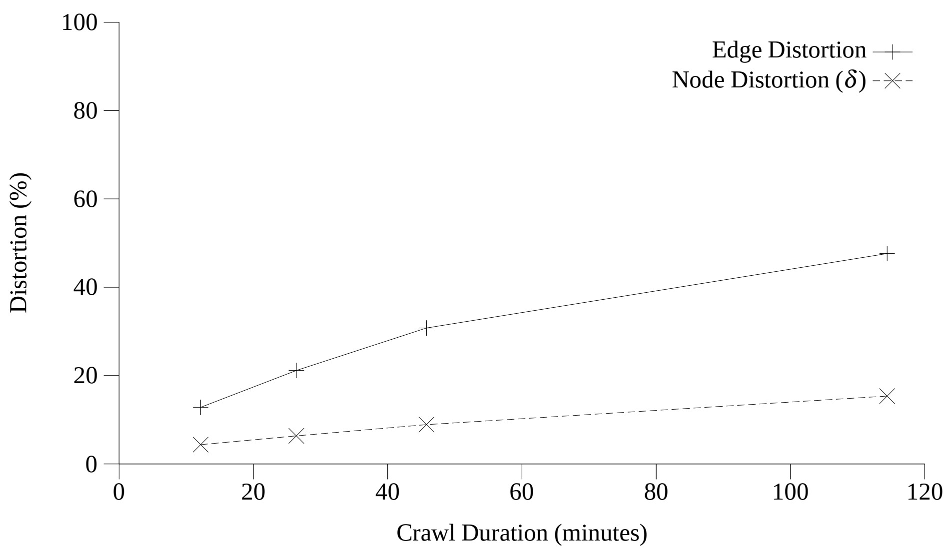


Fig. 3: Effect of crawl speed on the accuracy of captured snapshots

5 Sampling

While fast, Cruiser unavoidably takes $O(|V|)$ time, which means it may still be too slow to capture accurate snapshots of system with a very large population (V) or when the per-peer state is time-consuming to collect. For such cases, we need a mechanism to collect unbiased samples, which is the topic of the this chapter.

5.1 Sampling with Dynamics

We develop a formal and general model of a P2P system as follows. If we take an instantaneous snapshot of the system at time t, we can view the overlay as a graph $G(V,E)$ with the peers as vertices and connections between the peers as edges. Extending this notion, we incorporate the dynamic aspect by viewing the system as an infinite set of time-indexed graphs, $G_t = G(V_t, E_t)$. The most common approach for sampling from this set of graphs is to define a measurement window, $[t_0, t_0 + \Delta]$, and select peers uniformly at random from the set of peers who are present at any time during the window: $V_{t_0,t_0+\Delta} = \bigcup_{t=t_0}^{t_0+\Delta} V_t$. Thus, it does not distinguish between occurrences of the same peer at different times.

This approach is appropriate if peer session lengths are exponentially distributed (i.e., memoryless). However, existing measurement studies [18,38,42,46] show session lengths are heavily skewed, with many peers being present for just a short time (a few minutes) while other peers remain in the system for a very long time (i.e., longer than Δ). As a consequence, as Δ increases, the set $V_{t_0,t_0+\Delta}$ includes an increasingly large fraction of short-lived peers.

A simple example may be illustrative. Suppose we wish to observe the number of files shared by peers. In this example system, half the peers are up all the time and have many files, while the other peers remain for around 1 minute and are immediately replaced by new short-lived peers who have few files. The technique used by most studies would observe the system for a long time (Δ) and incorrectly conclude that most of the peers in the system have very few files. Moreover, their results will depend on how long they observe the system. The longer the measurement window, the larger the fraction of observed peers with few files.

One fundamental problem of this approach is that it focuses on sampling *peers* instead of *peer properties*. It selects each sampled vertex at most once. However, the property at the vertex may change with time. Our goal should not be to select a vertex $v_i \in \bigcup_{t=t_0}^{t_0+\Delta} V_t$, but rather to sample the property at v_i at a particular instant t. Thus, we distinguish between occurrences of the same peer at different times: samples $v_{i,t}$ and $v_{i,t'}$ gathered at distinct times $t \neq t'$ are viewed as distinct, even when they come from the same peer. *The key difference is that it must be possible to sample from the same peer more than once, at different points in time.* Using the formulation $v_{i,t} \in V_t$, $t \in [t_0, t_0 + \Delta]$, the sampling technique will not be biased by the dynamics of peer behavior, because the sample set is decoupled from peer session lengths. To our knowledge, no prior P2P measurement studies relying on sampling make this distinction.

Returning to our simple example, our approach will correctly select long-lived peers half the time and short-lived peers half the time. When the samples are examined, they will show that half of the peers in the system at any given moment have many files while half of the peers have few files, which is exactly correct.

If the measurement window (Δ) is sufficiently small, such that the distribution of the property under consideration does not change significantly during the measurement window, then we may relax the constraint of choosing t uniformly at random from $[t_0, t_0 + \Delta]$.

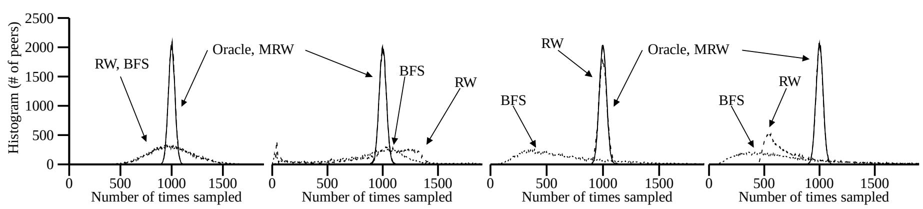
We still have the significant problem of selecting a peer uniformly at random from those present at a particular time. We begin to address this problem in the next section.

	Erdös–Rényi	Gnutella	Watts–Strogatz	Barabási–Albert
Breadth-First Search	$4.54 \cdot 10^{-4}$	$2.73 \cdot 10^{-3}$	4.73^{-3}	$2.77 \cdot 10^{-3}$
Random Walk	$3.18 \cdot 10^{-4}$	$1.57 \cdot 10^{-3}$	7.64^{-5}	$2.84 \cdot 10^{-3}$
Metropolis–Hastings	$5.97 \cdot 10^{-5}$	$5.79 \cdot 10^{-5}$	6.08^{-5}	$5.22 \cdot 10^{-5}$

Table 5: Kolmogorov-Smirnov test statistic for techniques over static graphs. Values above $1.07 \cdot 10^{-4}$ lie in the rejection region at the 5% level.

5.2 Sampling from Static Graphs

We now turn our attention to topological causes of bias. Towards this end, we momentarily set aside the temporal issues by assuming a static, unchanging graph. The selection process begins with knowledge of one peer (vertex) and progressively queries peers for a list of neighbors. The goal is to select peers uniformly at ran-



(a) Erdös–Rényi **(b)** Gnutella **(c)** Watts–Strogatz (small world) **(d)** Barabási–Albert (power-law)

Fig. 4: Bias of different sampling techniques; after collecting $k \cdot |V|$ samples. The figures show how many peers (y-axis) were selected x times.

dom. In any graph-exploration problem, we have a set of visited peers (vertices) and a front of unexplored neighboring peers. There are two ways in which algorithms differ: *(i)* how to chose the next peer to explore, and *(ii)* which subset of the explored peers to select as samples. Prior studies use simple breadth-first or depth-first approaches to explore the graph and select all explored peers. These approaches suffer from several problems:

- The discovered peers are correlated by their neighbor relationship.
- Peers with higher degree are more likely to be selected.
- Because they never visit the same peer twice, they will introduce bias when used in a dynamic setting as described in Section 5.1.

5.2.1 Random Walks

A better candidate solution is the random walk, which has been extensively studied in the graph theory literature (for an excellent survey see [32]). We briefly summarize the key terminology and results relevant to sampling. The transition matrix $P(x,y)$ describes the probability of transitioning to peer y if the walk is currently at peer x:

$$P(x,y) = \begin{cases} \frac{1}{\text{degree}(x)} & y \text{ is a neighbor of x,} \\ 0 & \text{otherwise} \end{cases}$$

If the vector v describes the probability of currently being at each peer, then the vector $v' = vP$ describes the probability after taking one additional step. Likewise, vP^r describes the probability after taking r steps. As long as the graph is connected and not bipartite, the probability of being at any particular node, x, converges to a *stationary distribution*:

$$\pi(x) = \lim_{r\to\infty}(vP^r)(x) = \frac{\text{degree}(x)}{2\cdot|E|}$$

In other words, if we select a peer as a sample every r steps, for sufficiently large r, we have the following good properties:

- The information stored in the starting vector, v, is lost, through the repeated selection of random neighbors. Therefore, there is no correlation between selected peers. Alternately, we may start many walks in parallel. In either cases, after r steps, the selection is independent of the origin.
- While the stationary distribution, $\pi(x)$, is biased towards peers with high degree, the bias is precisely known, allowing us to correct it.
- Random walks may visit the same peer twice, which lends itself better to a dynamic setting as described in Section 5.1.

In practice, r need not be exceptionally large. For graphs where the edges have a strong random component (e.g., small-world graphs such as peer-to-peer networks), it is sufficient that the number of steps exceed the log of the population size, i.e., $r \geq O(\log|V|)$.

5.2.2 Adjusting for degree bias

To correct the bias towards high degree peers, we make use of the Metropolis–Hastings method for Markov Chains. Random walks on a graph are a special case of Markov Chains. In a regular random walk, the transition matrix $P(x,y)$ leads to the stationary distribution $\pi(x)$, as described above. We would like to choose a new transition matrix, $Q(x,y)$, to produce a different stationary distribution, $\mu(x)$. Specifically, we desire $\mu(x)$ to be the uniform distribution so that all peers are equally likely to be at the end of the walk. Metropolis–Hastings [6, 16, 36] provides us with the desired $Q(x,y)$:

$$Q(x,y) = \begin{cases} P(x,y)\min\left(\frac{\mu(y)P(y,x)}{\mu(x)P(x,y)},1\right) & \text{if } x \neq y, \\ 1-\sum_{z\neq x} Q(x,z) & \text{if } x = y \end{cases}$$

Equivalently, to take a step from peer x, select a neighbor y of x as normal (i.e., with probability $P(x,y)$). Then, with probability $\min\left(\frac{\mu(y)P(y,x)}{\mu(x)P(x,y)},1\right)$, accept the move. Otherwise, return to x (i.e., with probability $1-\sum_{z\neq x} Q(x,z)$).

To collect uniform samples, we have $\frac{\mu(y)}{\mu(x)} = 1$, so the move-acceptance probability becomes:

$$\min\left(\frac{\mu(y)P(y,x)}{\mu(x)P(x,y)},1\right) = \min\left(\frac{\text{degree}(x)}{\text{degree}(y)},1\right)$$

Therefore, our algorithm for selecting the next step from some peer x is as follows:

- Select a neighbor y of x uniformly at random.
- Query y for a list of its neighbors, to determine its degree.
- Generate a random value, p, uniformly between 0 and 1.
- If $p \leq \frac{\text{degree}(x)}{\text{degree}(y)}$, y is the next step.
- Otherwise, remain at x as the next step.

We call this the Metropolized Random Walk (MRW). Qualitatively, the effect is to suppress the rate of transition to peers of higher degree, resulting in selecting each peer with equal probability.

5.2.3 Evaluation

Although [6] provides a proof of correctness for the Metropolis–Hastings method, to ensure the correctness of our implementation we conduct evaluations through simulation over static graphs. This additionally provides the opportunity to compare MRW with conventional techniques such as Breadth-First Search (BFS) or naive random walks (RW) with no adjustments for degree bias.

To evaluate a technique, we use it to collect a large number of sample vertices from a graph, then perform a goodness-of-fit test against the uniform distribution. For Breadth-First Search, we simulate typical usage by running it to gather a batch of 1,000 peers. When one batch of samples is collected, the process is reset and begins anew at a different starting point. To ensure robustness with respect to different kinds of connectivity structures, we examine each technique over several types of graphs as follows:

- **Erdös–Rényi:** The simplest variety of random graphs
- **Watts–Strogatz:** "Small world" graphs with high clustering and low path lengths
- **Barabási–Albert:** Graphs with extreme degree distributions, also known as power-law or scale-free graphs
- **Gnutella:** Snapshots of the Gnutella ultrapeer topology, captured in our earlier work [48]

To make the results more comparable, the number of vertices ($|V| = 161,680$) and edges ($|E| = 1,946,596$) in each graph are approximately the same.[2] Table 5 presents the results of the goodness-of-fit tests after collecting $1000 \cdot |V|$ samples, showing that Metropolis–Hastings appears to generate uniform samples over each type of graph, while the other techniques fail to do so by a wide margin.

Figure 4 explores the results visually, by plotting the number of times each peer is selected. If we select $k \cdot |V|$ samples, the typical node should be selected k times, with other nodes being selected close to k times approximately following a normal distribution with variance k.[3] We used $k = 1,000$ samples. We also include an "Oracle" technique, which selects peers uniformly at random using global information. The Metropolis–Hastings results are virtually identical to the Oracle, while the other techniques select many peers much more and much less than k times. In the Gnutella, Watts–Strogatz, and Barabási–Albert graphs, Breadth-First Search exhibits a few vertices that are selected a large number of times ($> 10,000$). The (not-adjusted) Random Walk (RW) method has similarly selected a few vertices an exceptionally large number of times in the Gnutella and Barabási–Albert models. The Oracle and MRW, by contrast, did not select any vertex more than around 1,300 times.

In summary, the Metropolis–Hastings method selects peers uniformly at random from a static graph. The next section examines the additional complexities when selecting from a dynamic graph, introduces appropriate modifications, and evaluates the algorithm's performance.

5.3 Empirical Results

In addition to the simulator version, we have implemented the MRWB algorithm for sampling from real peer-to-peer networks into a tool called `ion-sampler`. The

[2] Erdös–Rényi graphs are generated based on some probability p that any edge may exist. We set $p = \frac{2|E|}{|V|\cdot(|V|-1)}$ so that there will be close to $|E|$ edges, though the exact value may vary slightly. The Watts–Strogatz model require that $|E|$ be evenly divisible by $|V|$, so in that model we use $|E| = 1,940,160$.

[3] Based on the normal approximation of a binomial distribution with $p = \frac{1}{|V|}$ and $n = k|V|$.

following subsections briefly describe the implementation and usage of `ion-sampler` and present empirical experiments to validate its accuracy.

5.3.1 Ion-Sampler

The `ion-sampler` tool uses a modular design that accepts plug-ins for new peer-to-peer systems.[4] A plug-in can be written for any peer-to-peer system that allows querying a peer for a list of its neighbors. The `ion-sampler` tool hands IP-address:port pairs to the plug-in, which later returns a list of neighbors or signals that a timeout occurred. The `ion-sampler` tool is responsible for managing the walks. It outputs the samples to standard output, where they may be easily read by another tool that collects the actual measurements. For example, `ion-sampler` could be used with existing measurement tools for measuring bandwidth to estimate the distribution of access link bandwidth in a peer-to-peer system. Listing 1 shows an example of using `ion-sampler` to sample peers from Gnutella.

5.3.2 Empirical Validation

The topology snapshots from Cruiser provide a point of reference for the degree distribution to evaluate the accuracy of `ion-sampler` empirically. By capturing every peer, Cruiser is immune to sampling difficulties. However, because the network changes as Cruiser operates, its snapshots are slightly distorted. In particular, peers arriving near the start of the crawl are likely to have found additional neighbors by the time Cruiser contacts them. Therefore, we intuitively expect a slight upward bias in Cruiser's observed degree distribution. For this reason, we would not expect a perfect match between Cruiser and sampling, but if the sampling is unbiased we still expect them to be very close. We can view the CCDF version of the degree distribution captured by Cruiser as a close upper-bound on the true degree distribution.

Figure 5 presents a comparison of the degree distribution of reachable ultrapeers in Gnutella, as seen by Cruiser and by the sampling tool (capturing approximately 1,000 samples with $r = 25$ hops). It also includes the results of a short crawl,[5] a sampling technique commonly used in earlier studies (e.g., [42]). We interleaved running these measurement tools to minimize the change in the system between measurements of different tools, in order to make their results comparable.

Examining Figure 5, we see that the full crawl and sampling distributions are quite similar. The sampling tool finds slightly more peers with lower degree, compared to the full crawl, in accordance with our expectations described above. We

[4] In fact, it uses the same plug-in architecture as our earlier, heavy-weight tool, Cruiser, which exhaustively crawls peer-to-peer systems to capture topology snapshots.

[5] A "short crawl" is a general term for a progressive exploration of a portion of the graph, such as by using a breadth-first or depth-first search. In this case, we randomly select the next peer to explore.

examined several such pairs of crawling and sampling data and found the same pattern in each pair. By comparison, the short crawl exhibits a substantial bias towards high degree peers relative to both the full crawl and sampling.

5.3.3 Efficiency

Having demonstrated the validity of the MRWB technique, we now turn our attention to its efficiency. Performing the walk requires $n \cdot r$ queries, where n is the desired number of samples and r is the length of the walk in hops. If r is too low, significant bias may be introduced. If r is too high, it should not introduce bias, but is less efficient. From graph theory, we expect to require $r \geq O(\log |V|)$ for an ordinary random walk. Based on our empirical experiments in [47], we conservatively

```
bash$ ./ion-sampler gnutella --hops 25 -n 10
10.8.65.171:6348
10.199.20.183:5260
10.8.45.103:34717
10.21.0.29:6346
10.32.170.200:6346
10.201.162.49:30274
10.222.183.129:47272
10.245.64.85:6348
10.79.198.44:36520
10.216.54.169:44380
bash$
```

Listing 1: Example usage of the `ion-sampler` tool. We specify that we want to use the Gnutella plug-in, each walk should take 25 hops, and we would like 10 samples. The tool then prints out 10 IP-address:port pairs. We have changed the first octet of each result to “10” for privacy reasons.

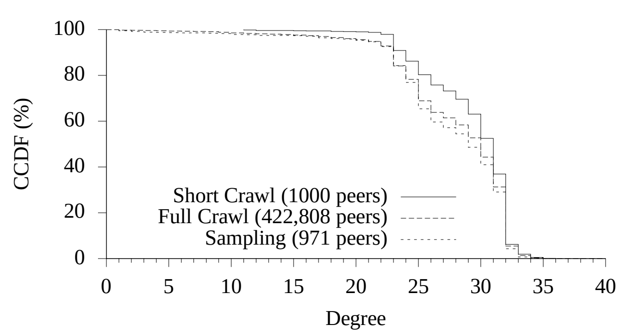


Fig. 5: Comparison of degree distributions observed from sampling versus exhaustively crawling all peers

regard a choice of $r = 25$ as a safe walk length for Gnutella. Choosing $r = 25$, we can collect 1,000 samples by querying 25,000 peers, over an order of magnitude in savings compared with performing a full crawl which must contact more than 400,000.

6 Summary and Future Work

The first half of this chapter surveys techniques for measuring attributes of P2P systems as well as characterizations derived from the application of those techniques. The second half explores two measurement techniques in detail—crawling and sampling—and demonstrates the importance of validating measurement methodology.

In our ongoing work, we are exploring different techniques to improve the accuracy and efficiency of the crawling and sampling technique presented here (and earlier presented in [47,48]). Additionally, we are examining large-scale traffic monitoring over Distributed Hash Tables (DHTs).

References

1. Adamic, L.A., Lukose, R.M., Huberman, B., Puniyani, A.R.: Search in power-law networks. Phys. Rev. E **64**(46135) (2001)
2. Adar, E., Huberman, B.A.: Free riding on Gnutella. First Monday **5**(10) (2000)
3. Annexstein, F.S., Berman, K.A., Jovanovic, M.A.: Latency effects on reachability in large-scale peer-to-peer networks. In: Symposium on Parallel Algorithms and Architectures, pp. 84–92. Crete, Greece (2001)
4. Bhagwan, R., Savage, S., Voelker, G.: Understanding availability. In: International Workshop on Peer-to-Peer Systems (2003)
5. Bustamante, F.E., Qiao, Y.: Friendships that last: Peer lifespan and its role in P2P protocols. In: International Workshop on Web Content Caching and Distribution (2003)
6. Chib, S., Greenberg, E.: Understanding the Metropolis–Hastings algorithm. The American Statistician **49**(4), 327–335 (1995)
7. Chu, J., Labonte, K., Levine, B.N.: Availability and locality measurements of peer-to-peer file systems. In: ITCom: Scalability and Traffic Control in IP Networks II Conferences (2002)
8. Clip2.com, Inc.: Gnutella: To the bandwidth barrier and beyond (2000)
9. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: International Conference on Distributed Computing Systems (2002)
10. Erman, D., Ilie, D., Popescu, A., Nilsson, A.A.: Measurement and analysis of BitTorrent signaling traffic. In: Nordic Teletraffic Seminar. Oslo, Norway (2004)
11. Fessant, F.L., Handurukande, S., Kermarrec, A.M., Massoulie, L.: Clustering in peer-to-peer file sharing workloads. In: International Workshop on Peer-to-Peer Systems (2004)
12. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks. In: IEEE INFOCOM (2004)
13. Guha, S., Daswani, N., Jain, R.: An experimental study of the Skype peer-to-peer VoIP system. In: International Workshop on Peer-to-Peer Systems. Santa Barbara, CA, USA (2006)
14. Gummadi, K.P., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: ACM Symposium on Operating Systems Principles (2003)

15. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Zhang, X.: Measurements, analysis, and modeling of BitTorrent-like systems. In: Internet Measurement Conference. Berkeley, CA (2005)
16. Hastings, W.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**, 97–109 (1970)
17. He, Q., Ammar, M.: Congestion control and message loss in Gnutella networks. In: Multimedia Computing and Networking. Santa Clara, CA (2004)
18. Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P.A., Hamra, A.A., Garces-Erice, L.: Dissecting BitTorrent: Five months in a torrent's lifetime. In: Passive and Active Measurement Workshop (2004)
19. Jiang, S., Zhang, X.: Floodtrail: An efficient file search technique in unstructured peer-to-peer systems. In: Globecom. San Francisco, CA (2003)
20. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is P2P dying or just hiding? In: Globecom. Dallas, TX (2004)
21. Karbhari, P., Ammar, M., Dhamdhere, A., Raj, H., Riley, G., Zegura, E.: Bootstrapping in Gnutella: A measurement study. In: Passive and Active Measurement Workshop (2004)
22. Klemm, A., Lindemann, C., Vernon, M., Waldhorst, O.P.: Characterizing the query behavior in peer-to-peer file sharing systems. In: Internet Measurement Conference. Taormina, Italy (2004)
23. Leibowitz, N., Bergman, A., Ben-Shaul, R., Shavit, A.: Are file swapping networks cacheable? Characterizing P2P traffic. In: International Web Caching Workshop (2002)
24. Leibowitz, N., Ripeanu, M., Wierzbicki, A.: Deconstructing the Kazaa network. In: IEEE Workshop on Internet Applications (2003)
25. Leonard, D., Rai, V., Loguinov, D.: On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. In: SIGMETRICS (2005)
26. Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-efficient management of DHT routing tables. In: USENIX/ACM Symposium on Networked Systems Design and Implementation. Boston, MA (2005)
27. Liang, J., Kumar, R., Ross, K.W.: The Kazaa overlay: A measurement study. Computer Networks Journal (Elsevier) (2005)
28. Liang, J., Kumar, R., Xi, Y., Ross, K.W.: Pollution in P2P file sharing systems. In: IEEE INFOCOM. Miami, FL (2005)
29. Liu, Y., Liu, X., Xiao, L., Ni, L.M., Zhang, X.: Location-aware topology matching in P2P systems. In: IEEE INFOCOM (2004)
30. Liu, Y., Zhuang, Z., Xiao, L., Ni, L.M.: AOTO: Adaptive overlay topology optimization in unstructured P2P systems. In: Globecom. San Francisco, CA (2003)
31. Liu, Y., Zhuang, Z., Xiao, L., Ni, L.M.: A distributed approach to solving overlay mismatching problem. In: International Conference on Distributed Computing Systems (2004)
32. Lovász, L.: Random walks on graphs: A survey. Combinatorics: Paul Erdös is Eighty **2**, 1–46 (1993)
33. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: International Conference on Supercomputing (2002)
34. Lv, Q., Ratnasamy, S., Shenker, S.: Can heterogeneity make Gnutella scalable? In: International Workshop on Peer-to-Peer Systems (2002)
35. Markatos, E.P.: Tracing a large-scale peer to peer system: an hour in the life of Gnutella. In: CC Grid (2002)
36. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equations of state calculations by fast computing machines. J. of Chemical Physics **21**, 1087–1092 (1953)
37. Plissonneau, L., Costeux, J.L., Brown, P.: Analysis of peer-to-peer traffic on ADSL. In: Passive and Active Measurement Workshop, pp. 69–82. Boston, MA (2005)
38. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The BitTorrent P2P file-sharing system: Measurements and analysis. In: International Workshop on Peer-to-Peer Systems. Ithaca, NY (2005)
39. Rhea, S., Geels, D., Kubiatowicz, J.: Handling churn in a DHT. In: USENIX, pp. 127–140 (2004)

40. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. IEEE Internet Comput. **6**(1) (2002)
41. Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., Levy, H.M.: An analysis of Internet content delivery systems. In: Symposium on Operating Systems Design and Implementation, pp. 315–327 (2002)
42. Saroiu, S., Gummadi, P.K., Gribble, S.D.: Measuring and analyzing the characteristics of Napster and Gnutella hosts. Multimedia Systems J. **9**(2), 170–184 (2003)
43. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. IEEE/ACM Trans. Netw. **12**(2), 219–232 (2004)
44. Sripanidkulchai, K.: The popularity of Gnutella queries and its implications on scalability (2001). URL `http://www-2.cs.cmu.edu/~kunwadee/research/p2p/paper.html`
45. Stutzbach, D., Rejaie, R.: Improving lookup performance over a widely-deployed DHT. In: IEEE INFOCOM. Barcelona, Spain (2006)
46. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: Internet Measurement Conference. Rio de Janeiro, Brazil (2006)
47. Stutzbach, D., Rejaie, R., Duffield, N., Sen, S., Willinger, W.: On unbiased sampling for unstructured peer-to-peer networks. IEEE/ACM Trans. Netw. (To appear)
48. Stutzbach, D., Rejaie, R., Sen, S.: Characterizing unstructured overlay topologies in modern p2p file-sharing systems. IEEE/ACM Trans. Netw. **16**(2) (2007)
49. Stutzbach, D., Zhao, S., Rejaie, R.: Characterizing files in the modern Gnutella network. Multimedia Systems J. **13**(1) (2007)
50. Wang, C., Xiao, L., Liu, Y., Zheng, P.: Distributed caching and adaptive search in multilayer P2P networks. In: International Conference on Distributed Computing Systems (2004)
51. Yang, M., Zhang, Z., Li, X., Dai, Y.: An empirical study of free-riding behavior in the Maze P2P file-sharing system. In: International Workshop on Peer-to-Peer Systems. Ithaca, NY (2005)